This article can be cited as M. Wagarachchi and A. Karunananda, Optimization of Artificial Neural Network Architecture Using Neuroplasticity, International Journal of Artificial Intelligence, vol. 15, no. 1, pp. 112-125, 2017.

Optimization of Artificial Neural Network Architecture Using Neuroplasticity

Mihirini Wagarachchi[1] and **Asoka Karunananda**[2]

[1] Faculty of Engineering, University of Ruhuna
Sri Lanka
Email: mihirini@is.ruh.ac.lk

[2]Faculty of Information Technology, University of Moratuwa
Sri Lanka
Email: asoka@mrt.ac.lk

ABSTRACT

Artificial neural networks, which are inspired by the behavior of central nervous system have the capability of finding good generalized solutions for many real world problems due to their characteristics such as massively parallel, ability to learn and adapt to the environment by altering the synaptic weights. However, despite of all the advantages of artificial neural networks, determining the most appropriate architecture for the given problem still remains as an unsolved problem. This paper presents a pruning method based on the backpropagation algorithm to solve this problem. The pruning method is inspired by the concepts of neuroplasticity and experimental results show that the proposed method approaches the minimal architecture faster than the other existing methods.

Keywords: artificial neural networks, backpropagation, delta values, hidden layer architecture, neuroplasticity, pruning.

Mathematics Subject Classification: 68T05, 82C32, 92B20

Computing Classification System: I.6.5

1. INTRODUCTION

The artificial neural network is a computational model, which mimics the behavior of the human's central nervous system. The major unit of the central nervous system, the brain, consists of an innumerous number of small functional units called neurons, which are interconnected by dendrites and axons. When a dendrite receives a signal from the environment, its electrical potential changes and attains to a threshold and emits an electrical pulse thorough axon to several other neurons. Likewise, artificial neural networks are created to do similar process and they are made by computational units called artificial neurons. In general, it assumes that neurons in artificially designed

network lie on layers and each network contains an input layer, an output layer and the number of hidden layers. These stuctures are massively parallel and highly complex structure which are capable to grab the knowledge from its environment and interconnected strengths, known as synaptic weights are used to store the grabbed knowledge. So that as in human brain, neural networks are also capable of adaptive, input-output mapping, evidential response and fault tolerance (Haykin, n.d.-b).

In spite of many advantages, the deciding the most appropriate neural architecture for any specific task still remains as an unsolved problem. It has been observed that both too large and too small architectures show advantages as well as disadvantages (Castellano et al., 1997). When network is too large it learns fast with avoiding local minima (Rumelhart, Geoffrey, & Williams, 1986) (Plaut & Hinton, 1987). In (Yu, 1992) it is theoretically explained that local minima problem obtained by backpropagation algorithm can be minimized when number of hidden units equal to the number of training patterns. Also large networks can form complex decision regions as problem requires (Lippmann, 1987) and shows better fault tolerance in damage conditions. However, when there are too many parameters generalization ability declines as it fails to distinguish similar neurons. On the other hand, networks with too few parameters show better generalization but neurons in these networks may not learn data properly (LeCun, 1989), (Denker et al., 1987), (Reed, 1993), (Aran et al., 2009), (Setiono & Liu, 1995).

To solve this problem different approaches have been used. The algorithms known as pruning algorithms start with an over-sized network and eliminate unimportant neurons until the optimal network occurs(LeCun et al., 1989),(Kamruzzaman & Hasan, 2010). In contrast, algorithms based on constructive methods are starting with minimal network and add neurons and connection weights until it reaches to the optimal solution (Ash, 1989), (Fahlman & Lebiere, 1989) The main disadvantage constructive method is initial small networks easily stuck with local minima and then training time may increase. Some researchers have been used pruning-constructive hybrid algorithms to overcome the problem of hidden layer architecture (Islam & Murase, 2001).

This paper presents a pruning algorithm to determine the most befitting topology for hidden layers in neural network based on backpropagation algorithm. The proposed method is hypnotized by concepts of neuroplasticity. The process begins with a oversized network trained by backpropagation (Rumelhart et al., 1986) algorithm. Firstly, it determines the number of hidden layers for the most appropriate network by using a bi-search algorithm (Niemann, n.d.) and then eliminate unimportant nodes. The *delta values* obtain by backpropagation algorithm are used to identify weak neurons.

The previous works are discussed in the next section, and Section 3 presents the method of determining the number of layers and recognizing the removable nodes. Section 4 discusses the new algorithm while the experiments and presents results in Section 5. Finally, conclusions are given in Section 6.

2. PREVIOUS WORKS

In this section, we discuss some works carried by other researchers and authors well on optimization of hidden layer architecture in artificial neural networks.

2.1. Pruning Algorithms

Pruning algorithms make network smaller by eliminating unnecessary weights or nodes. So that it is enable to reduce the cost of network while improve the generalization.

Le Cun et al (LeCun et al., 1989) proposed method called 'Optimal Brain Damage' to eliminate unimportant neurons from the network by measuring 'saliency' by using the second derivative of the error with respect to the connection weights. The main objective of algorithm is to find parameters whose removal will cause to minimize the error. When network is large Hessian matrix (Haykin, n.d.-a) becomes enormous. Hence, authors assume that the matrix is diagonal. By presenting 'Optimal Brain Surgeon' Hassibi et al (Hassibi & Stork, 1993) argue that Hessian matrix is strongly non diagonal for all the considered networks and hence, it may eliminate incorrect weights. However, optimal brain damage method also stuck with complex computations specially when working with inverse of the Hessian matrix.

G. Castellano et al (Castellano et al., 1997) proposed a method to iteratively prune hidden neurons from a feed-forward neural network by solving a system of linear in the least square sense using pre-conditioned conjugate gradient procedure. This algorithm has applied to solve some problems. Nevertheless, when for large network matrix corresponds to system of linear equations may have deficiency rank and hence, infinite number of solution may occur.

Augasta et al (Augasta & Kathirvalavakumar, 2011) applied pruning method called Neural Network Pruning by Significance (N2PS) by defining the significance which is computed by sigmoidal activation of a neuron as the sum-norm of its output and identify removable neurons by comparing this significant with a pre decided threshold value.

Authors presented an algorithm to determine the hidden layer architecture by using a pruning algorithm previously (N. M. Wagarachchi & Karunananda, 2013b), (N. Mihirini Wagarachchi & Karunananda, 2014). The proposed algorithm firstly decides the number of hidden layers in the most appropriate network by using accuracy factor defined by

$$AF = \frac{MR}{\sqrt{E}} \tag{1}$$

where MR denotes the generalization power and E is the error at the output layer. Although this algorithm chooses the number of hidden layers in the most appropriate network very correctly, it

needs to train the network several times by using backpropagation algorithm. So that it needs to put much effort to obtain the most appropriate network.

2.2. Constructive Algorithms

In constructive neural networks, the network structure is built during the training process by adding hidden layers, nodes and connections to a minimal neural network architecture. However, determining whether a weight should be added and it adds to the existing hidden layer of new layer is complex. Therefore in most algorithms, pre-defined and fixed number of nodes are added in the first hidden layer and the same number of nodes are added to second layer and so on (Sharma & Chandra, 2010). This number is crucial for the better performance of the network and it makes as small as possible to avoid the complicated computations during the training.

The cascade correlation algorithm (CCA), firstly proposed by (Fahlman & Lebiere, 1989). CCA is a well known and mostly used constructive algorithm. This algorithm has proposed as a solution of problems such as local minima problem.

The dynamic node creation (DNS) algorithm (Ash, n.d.) is supposed to be the first constructive algorithm for designing single layer feed-forward networks dynamically. Sridar et al (Sridhar & Ponnavaikko, 2011) improved the adaptive learning algorithm for multi-category tiling constructive neural networks for pattern classification problems.
Nevertheless, there are several such approaches; they are confined to single hidden layer networks or networks with a small number of hidden neurons and they do not match with the theoretical background as human brain consists of large enormous number of neurons and synaptic connections.

3. NEUROPLASTICITY APPROACH ON ARTIFICIAL NEURAL NETWORKS

This research is inspired by the fact that the nature has always over estimated. In the same line neuroscientists claim that while part of a body is paralyzed, another part could be maximized their function to compensate the damaged neurons. So that artificial neural networks are modelled to mimic the functional behaviour of human brain (Sun, 2008) (Bringsjord, 2008) (Jain, Mao, & Mohiuddinis , 1996). Until recently, scientists and the philosophers in the field of neuroscience worked with the notion that the human brain is immutable and hard wired. It was postulated that no new neurons are born and functions of brain structures are fixed (Vollmer, 2010). The recent studies show that these assumptions are no longer correct and brain functions change throughout one's life. The change of brain neurons and its pathways to adapt to the surrounding environment is called 'neuroplasticity' and also referred as the 'brain plasticity' (Kuo, 2007) (Kleiner, 2011) (Kolb et al., 2003).

The structural changes in human brain can be occurred due to the various types of behaviour of neurons such as neurogenesis, neural migration, neural cell death, synapto genesis and synaptic

pruning (Stiles & Jernigan, 2010) (Stiles, Brown, Haist, & Jernigan, 2015). Moreover, Adapting to the surrounding by changing its pathways and functional abilities is one of the most fascinating features in human brain. These changes happens almost all the cortex and region in the brain and they can occur as Adaptive-dependent plasticity (Fu & Zuo, 2011), competitive plasticity , positive and negative plasticity.

3.1. Synaptic Pruning

The elimination of unnecessary synapses from the central nervous system is known as synaptic pruning (Santos & Noggle, 2011). Although this process last through the life span, the majority of the synapses eliminates from the human brain between the childbirth and the puberty. At the birth, human brain consists of more than 80 billion of neurons. During the first two years after the child's birth size of the brain grows significantly. In this period, there is no much neurogenesis take place. The growth of the brain occurs as the result of creation of new synapses and myelination of nervous fiber. Myelination refers the forming white substance surrounding the axon. Creation of new synapses is called the synaptogenesis. At the child's birth, generally a neuron consists of 2,500 connections. At 2 years, it becomes about 15,000 and this is far more than the functionally needed.

When synaptogenesis reaches peak level it starts to prune weak and unnecessary synapses from the central nervous system. Pruning occurs due to environmental factors and learning. While infant learning weak synapses eliminate by strengthening the functions of the remaining ones. Pruning process last until the death of healthy persons, but significantly occur until the adolescence. At the end of this process, brain contains about 50% synapses that were in two-year-old child.

3.2. Artificial Neural Networks and the Human Brain

Human brain which is having a phenomenal power is the most complex organ in the human body and known as a massively parallel and highly complex information-procession structure (Haykin, n.d.-b). The extraordinary power of human brain is far beyond that of any supercomputer today. The mechanism of human brain is far different from the conventional 'Von Neumann' architectural computer, which works gradually, sequentially through an algorithm. Among a big crowd in a town, we can recognize a friend, or identify a known voice in a noisy place. Is there any machine to model such complex behaviour? To answer this question artificial neural networks are developed by mimicking the some of the fascinating and remarkable features of the human brain.

In the human brain, dendrites, which project from the cell body or soma, receive signals and pass them to another cell body through an axon. When accumulated signals in cell body reach to a certain threshold limit, the neuron fires and electrical impulses pass through the axon. At the end, each axon is branched into number of synaptic knobs, also known as axon terminals. With synaptic connects, it

connects to other neighboring neurons and signal passes to those adjacent neurons through the synapses. Some synapses get positive outcomes from dendrites and they influence neurons to fire while some get negative outcomes and they weaken the signals. Approximately, a single neuron connects to 10^5 synapses and about 10^{16} synaptic connections.

Artificial neural networks are created to model this complex functional behaviour of the human brain by directly transferring the concepts of neurons. Neurons are represented by nodes or artificially designed neurons. The axons are corresponding to the connections between neurons. Dendrites and cell body are represented by connection weights and activation functions respectively. The synaptic weights of artificial neural networks represent the synapses of central nervous system.

The concept of training of artificial neural networks came from the psychologist Donald O. Hebbs famous theory "*When an axon of cell A is near enough to excite cell B or repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased*" (Hebb, n.d.).

However, it is still challenging to model human brain artificially. The Biological neurons and neuronal activity are absolutely complex than artificially created neurons. Generally, neurons in human brain do not simply sum the weighted inputs and the dendritic mechanisms in biological systems are much more perfected. Also, real neurons do not stay on until the inputs change and the outputs may encode information using complex pulse arrangements ("Overfitting," 2016).

4. DETERMINING THE HIDDEN LAYER ARCHITECTURE

Determining the hidden layer architecture is a great challenge in artificial neural networks. Although there are several approaches on hidden layer architecture, we find that available methods have various shortcomings. Therefore, still the problem of hidden layers remaining as an unsolved problem.In this research, firstly we propose an algorithm to determine the number of hidden layers in the most appropriate network and then discuss a pruning method to reach the optimal solution by removing unimportant nodes from each hidden layer.

4.1. Determining Number of Hidden Layers

The procedure of determining the number of hidden layers in the most appropriate network starts with a larger sized trained network. The main objective of this process is to reduce the number of layers in the given network where smaller sized network shows similar of better performance. We hypothesized that a large network can be reduced to a smaller sized network without degrading its performances by removing hidden layers and hidden neurons. The proposed algorithm works by comparing the generalization defined in eqn (3) corresponds to number of hidden layers of the middle value of the

array. According to the inequality holds, lower or upper part of the array eliminates and repeats the procedure until it reaches to the maximum value. The proposed algorithm is to search number of hidden layers H_n , which provides the maximum generalization $\varphi(H_n)$.

The algorithm begins with a network with H hidden layers, trained with backpropagation algorithm. Searching starts with middle element $\left[H/2\right]$ of the array $(1, 2, ..., H)$. Where [*N*] denotes the integer part of the number *N*. A binary comparison tree represents the procedure. Iteration continues by selecting alternatively middle elements of left and right parts of previously choose middle element. The process terminates when there is no integer between two values, which provide the highest generalization. The worst case arises when tree contains maximum branches. The number of iterations of the worst case is computed as

$$\frac{2\times \ln(H)}{\ln(2)} \tag{2}$$

It could be noted that various search algorithms (Ramírez-Ortegón, Märgner, Cuevas, & Rojas, 2013), (Precup, David, Petriu, Preitl, & Rădac, 2014) are available for parameter estimation problems. This algorithm is faster than the other available algorithms to peak search (Demaine, 2002) (Scholkmann, Boss, & Wolf, 2012) that reach to the peak by considering 3 consecutive numbers. Nevertheless, the proposed algorithm concerns the behaviour in the corresponding interval and hence, it reaches to target value with lesser iterations.

The proposed algorithm starts by a trained network with *H* hidden layers, and then compares the generalization of the middle value of the range and choose the network which gives the maximum generalization.

The generalization of the network for h hidden layers is given by

$$\varphi(h) \quad = \quad \frac{number\ of\ data\ which\ provides\ desired\ output}{total\ number\ of\ data\ in\ testing\ set}\times 100\% \tag{3}$$

The detailed algorithm is given below.

Input:

H: Number of hidden layers in the trained network

Output:

H_n: Number of hidden layers in the most appropriate network

1) Compute the generalization for *H* hidden layers, $\varphi(H)$,
2) Compute the generalization for 1 hidden layer, $\varphi(1)$
3) If $\varphi(1)\neq 100$, then $H_0 = 1$ and $H_n = H$

   else $H_n \quad = \quad 1$
4) Compute the middle value

$$H_1 = \left[\frac{H_0 + H_n}{2}\right]$$

5) If $\varphi(H_0) > \varphi(H_1) \geq \varphi(H_n)$, then $H_n = H_1$.

Else if $\varphi(H_0) \leq \varphi(H_1) < \varphi(H_n)$, then $H_0 = H_1$.

else when $\varphi(H_0) \leq \varphi(H_1) \geq \varphi(H_n)$,

compute $H_2 = \left[\frac{H_0 + H_1}{2}\right]$, and $H_3 = \left[\frac{H_1 + H_n}{2}\right]$

and then $H_0 = H_2$ and $H_n = H_3$

6) Repeat 4) and 5) until $H_n - H_0 . < 1$.

7) H_n is the desired number of layers in the most appropriate network.

4.2. Eliminate Unimportant Neurons

The traditional method of training a feed-forward artificial neural network is backpropagation algorithm that can be used successfully in many real world problems. The procedure starts with an over-sized network, which is trained by backpropagation algorithm. Number of hidden layers is decided by using the method discussed in previously. However, still this network contains some unnecessary neurons and this research uses the algorithm proposed by authors in (N. Mihirini Wagarachchi & Karunananda, 2014) and (N. M. Wagarachchi & Karunananda, 2013a) to eliminate those.

The basic idea behind the training network is to minimize the error (E) between desired output (d) and actual output (y). After n training cycles error $E(n)$ is given as

$$E(n) = \frac{1}{2}\sum_{j=1}^{m}\left(d_j(n) - y_j(n)\right)^2 \quad (4)$$

where d_j and y_j are desired and actual outputs of j^{th} neuron respectively. m is the number of neurons in the output layer. For N number of input/output training patterns, error becomes

$$E(n) = \frac{1}{2N}\sum_{k=1}^{N}\sum_{j=1}^{m}\left(d_{jk}(n) - y_{jk}(n)\right)^2 \quad (5)$$

The proposed algorithm prunes neurons as much as possible from the hidden layers of over-sized network while maintaining the same or better performance as the initial network. Pruning is done by using delta values of hidden layers. The delta value of the j^{th} neuron of the hidden layer immediately before the output layer is given by

$$\delta_i = f_h'(net_i)\sum_k \delta_k w_{ki} \quad (6)$$

where $f_h^{'}$ is the pre-defined activation function of the hidden layer, w_{ki} is the connection weight of the neurons i of the last hidden layer and neuron k of the output layer. δ_k refers the delta value of k^{th} neuron in the output layer define by

$$\delta_k = f_o^{'}(net_k)(d_k - y_k) \quad (7)$$

where $f_o^{'}$ is the activation function defined for output layer and d_k and y_k are the desired and actual outputs respectively. The intension of choosing a delta value is that the delta value of i^{th} neuron in last hidden layer at n^{th} training cycle $\delta_i^h(n)$ is calculated by using the error $e_k(n)$. While training the network connection weights are updated as follows.

$$w_{ki}^h(n+1) = w_{ki}^h(n) + \eta\delta_i^h(n) f_h(n) \quad (8)$$

where h is the number of hidden layers in the network and η is the learning rate. The above Eqn. (8) implies that zero delta value means there is no update of the particular weights. Therefore, hidden neurons with zero delta values are not contributed to decrease the error in the training process. So that the hidden neurons with zero delta values are identified as less salience neurons and elimination of them from the topology does not affect on the performance of the network.

Empirical results show that very often, there is a correlation between summation of delta values of hidden layers and the output error, which can be positive or negative. Thus, we use this correlation to identify the removable neurons to obtain a more precise network. Let this correlation be denoted by $\gamma_{\delta_h,E}$.

$$\gamma_{\delta_h,E} = corr\left(\sum_{k=1}^{n_h}\delta_k, E\right) \quad (9)$$

Therefore, to obtain optimal network, the correlation defined in the above is used. If the correlation is positive, minimal architecture obtains by removing neurons with positive delta that are very close to zero. In contrast, when the correlation is negative, neurons with large negative delta values remove to obtain the desired architecture.

The pruning neurons have the same meaning as the synaptic pruning in neuroscience. It facilitates changes in neural configuration by removing weak neurons and synapses while strengthening the remaining. As in synaptic pruning, while pruning the weak neurons and the synaptic connections from the network it merges the similar neurons to strengthen their functions.

4.3. Merge Similar Neurons

The entire process of pruning neurons is inspired by the concepts of neuroplasticity and synaptic pruning. When an infant is learning unnecessary neurons remove from the human brain while increasing the functions of the remaining. In similar, the process of the proposed algorithm maximizes the weights of synaptic connections while removing the unimportant neurons. However, the

contribution of these weights in the error decay process is not negligible. Hence, while removing these connection weights are merging with the similar neurons to obtain more efficient network.

Let k^{th} neuron of hidden layer h be identified as a removable neuron. Suppose $V = (v_{ij})_{q \times p}$ and $W = (w_{ij})_{r \times q}$ are input and output vectors of layer h respectively. When removing k^{th} neuron, the row vector $V^{R_k} = [v_{k1}, v_{k2}, ..., v_{kq}]$ and the column vector $W^{C_k} = [w_{1k}, w_{2k}, ..., w_{qk}]^T$ will be removed and while removing they merge with *similar vectors*. By similar vectors it means that the vectors with the same orientation. So that, when two vectors V^{R_i} and V^{R_k} are similar, if

$$\left\langle \frac{V^{R_i}}{\left\| V^{R_i} \right\|}, \frac{V^{R_k}}{\left\| V^{R_k} \right\|} \right\rangle = 1 \qquad (10)$$

Thus, if neuron *k* in layer h is identified as the removable neuron, and V^{R_i} and W^{C_j} are the similar vectors to V^{R_k} and W^{C_k} respectively then V^{R_k} merge with V^{R_i} and W^{C_k} merge with W^{C_j}.

5. EXPERIMENTS AND RESULTS

To test the performance of the proposed pruning method, several data sets with different input/output sizes were tested (*Table 1*). All the data sets are chosen from UCI Machinery repository ("UCI machine learning repository - Google Search," n.d.). Each data set divided into two classes namely testing and training. Initially, for each problem fully connected network created with H number of hidden layers and N hidden neuron and then trained with backpropagation algorithm. The log-sigmoid and linear function were used as activation functions for hidden layers and output layer respectively. The learning rate fixed as 0.1 for all the instances.

Table 1: The data sets

Training set	Input/output pattern	Initial Network	
		No. of Hidden Layers	No. of Hidden Neurons
Contraceptive	1473	12	1106
Iris	150	10	100
Gene	3175	12	2382
Cardio	2126	20	1590
Breast tissue	106	20	70
Knowledge	403	20	258

The number of hidden neurons in each layer is considered to be as $\left[N/H\right]$. For example, the data set of 'Contraceptive' has 1106 total number of hidden neurons. When there are 12 hidden layers, each layer contains $\left[1106/12\right]=92$ numbers of hidden neurons. When the number of hidden layers is 5, there are 221 hidden neurons in each layer. The algorithm presented in Section 4.1 was used to determine the number of hidden layers in the most appropriate network. *Table 2* shows the number of hidden layers that shows the best performance in each problem and generalization (accuracy) as a percentage for that particular number of layers. The experimental results show that most probably the best performance can achieve with $2-8$ hidden layers. Moreover, it is clear that single hidden layer architecture is not a solution, especially there are large numbers of hidden neurons. Nevertheless, *Iris* problem shows 100% accuracy for single hidden layer network, 3 layer network can train faster than it by maintaining the same performance.

Table 2 : Number of the hidden layers in the most appropriate network

Training set	No. of hidden layers in the most appropriate network	Accuracy %
Contraceptive	9	41.30
Iris	3	100.00
Gene	4	86.76
Cardio	4	80.23
Breast tissue	6	100.00
Knowledge	2	92.41

After determining the number of hidden layers of the best architecture, network trained by using the algorithm in Section 4.2 to eliminate insignificant hidden neurons. *Table 3* describes the performance of the new model. It is clear that the new algorithm reduces the size of network by improving the generalization power and limiting the number of training cycles. For example, in *Contraceptive* problem the number of hidden layers reduced from 1106 to 853 by improving accuracy of the training set by 23.25%. It can be observed that almost half of the neurons (49.11%) have been removed in the *Gene* problem.

Table 3 : Performance of the new model

Training set	New Model		Accuracy %
	No. of Hidden Layers	No. of Hidden Neurons	
Contraceptive	9	853	64.25
Iris	3	67	100.00
Gene	4	1212	88.13
Cardio	4	1356	91.03
Breast tissue	6	66	100.00
knowledge	2	108	100.00

In Section 3 we discussed that over pruning and under pruning of synapses cause some mental disorders. Similar happens in artificially designed networks. It shows poor measures in generalization and training cycles when over-pruning and under-pruning. For instance, in *Iris* problem 3 hidden

layers with 67 hidden neurons shows 100% accuracy. The hidden neurons are distributed as 27 – 22 – 18. Although the initial network shows same accuracy, the network is larger than the pruned network. However, if the pruning process continues, next step architecture becomes 22 – 19 – 12 with 98.13% accuracy. Thus, it shows that over pruning the topology causes the poor performance of the network.

6. CONCLUSIONS

This paper presents a pruning algorithm to obtain the optimal solution of hidden layer architecture in multilayered artificial neural networks. It hypothesized that any given large sized network can be reduced to a smaller sized network by reducing hidden layers and removing hidden neurons and the resultant network shows same or better performance. The most attractive features are, the method is simple and effective. There are no complex computations.

The backpropagation algorithm and modified version of it has been used in pruning. The experimental results show that any given network can be reduced to smaller network and the resultant network all most all the cases show better performance than the initial network. However, there are some limitations arise, especially when we consider networks with very large numbers of input/output training patterns.

REFERENCES

Aran, O., Yildiz, O. T., & Alpaydin, E. (2009). An incremental framework based on cross-validation for estimating the architecture of a multilayer perceptron. *International Journal of Pattern Recognition and Artificial Intelligence*, *23*(2), 159–190.

Ash, T. (1989). Dynamic node creation in backpropagation networks. *Connection Science*, *1*(4), 365–375.

Ash, T. (n.d.). Dynamic node creation in back-propagation networks. Technical Report 8901, Institute for Cognitive Science, University of California, San Diego, CA, USA.

Augasta, M. G., & Kathirvalavakumar, T. (2011). A novel pruning algorithm for optimizing feedforward neural network of classification problems. *Neural Processing Letters*, *34*(3), 241–258.

Bringsjord, S. (2008). Declarative/logic-based computational cognitive modeling. *The Handbook of Computational Cognitive Modeling, Cambridge University Press, Cambridge (Forthcoming)*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.5645&rep=rep1&type=pdf.

Castellano, G., Fanelli, A. M., & Pelillo, M. (1997). An iterative pruning algorithm for feedforward neural networks. *Neural Networks, IEEE Transactions on*, *8*(3), 519–531.

Demaine, E. (2002). *Introduction to Algorithms*. MIT Press, London. Retrieved from http://xcoursefiles.amirbarari.ir/designofalgorithms/SRL02/IntroductionToAlgorithms-FALL2005/SRL02-MIT-XEN-ITA-FALL2005-Lecture17-Shortest%20Paths%20I.pdf.

Denker, J., Schwartz, D., Wittner, B., Solla, S., Howard, R., & Jackel, L. (1987). Large automatic learning, rule extraction, and generalization. *Complex Systems*, *1*, 877–922.

Fahlman, S. E., & Lebiere, C. (1989). The cascade-correlation learning architecture. *Advances in Neural Information Processing Systems*, *2*, 524–532.

Fu, M., & Zuo, Y. (2011). Experience-dependent structural plasticity in the cortex. *Trends in Neurosciences*, *34*(4), 177–187.

Hassibi, B., & Stork, D. G. (1993). Second order derivatives for network pruning: Optimal Brain Surgeon. *Advances in Neural Information Processing Systems*, *5*(S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds.), 164–171.

Haykin, S. (n.d.-a). Multilayer Perceptrons. In *Neural Networks and Learning Machines*.

Haykin, S. (n.d.-b). *Neural Networks and Machine Learning* (Third Edition). Prentice Hall.

Hebb, D. O. The organization of behavior - Google Search. Retrieved from https://www.google.lk/?gws_rd=ssl#q=the+organization+of+behavior.

Islam, M. M., & Murase, K. (2001). A new algorithm to design compact two-hidden-layer artificial neural networks. *Neural Networks*, *14*(9), 1265–1278.

Jain, A. K., Mao, J., & Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. *Computer*, (3), 31–44.

Kamruzzaman, S. M., & Hasan, A. R. (2010). Pattern Classification using Simplified Neural Networks. *arXiv Preprint arXiv:1009.4983*. Retrieved from http://arxiv.org/abs/1009.4983

Kleiner, J. S. (2011). Brain Plasticity. *Encyclopedia of Clinical Neuropsychology*, 440–441.

Kolb, B., Gibb, R., & Robinson, T. E. (2003). Brain plasticity and behavior. *Current Directions in Psychological Science*, *12*(1), 1–5.

Kuo, M.-F. (2007). *Neuroplasticity: induction and modulation by external stimulation and pharmacological intervention*. Niedersächsische Staats-und Universitätsbibliothek Göttingen. Retrieved from https://www.deutsche-digitale-bibliothek.de/binary/KZL2STDDIXAN4DZB7X73PX6KCGWHBDPO/full/1.pdf.

LeCun, Y. (1989). Generalization and network design strategies. *Connectionism in Perspective*, Pfeifer, Schreter, Fogelman and Steels (Eds.), Elsevier, 143–155.

LeCun, Y., Denker, J. S., Solla, S. A., Howard, R. E., & Jackel, L. D. (1989). Optimal brain damage. In *NIPs* (Vol. 2, pp. 598–605). Retrieved from http://yann.lecun.com/exdb/publis/pdf/lecun-90b.pdf.

Lippmann, R. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine*, *4*(2), 4–22.

Niemann, T. (n.d.). Sorting and searching algorithms. *Epaperpress.com*. Retrieved from http://epaperpress.com/sortsearch/download/sortsearch.pdf.

Overfitting. (2016, June 8). In *Wikipedia, the free encyclopedia*. Retrieved from https://en.wikipedia.org/w/index.php?title=Overfitting&oldid=724236754.

Plaut, D. C., & Hinton, G. E. (1987). Learning sets of filters using back propagation. *Computer Speech and Language*, *2*, 35–61.

Precup, R.-E., David, R.-C., Petriu, E. M., Preitl, S., & Rădac, M.-B. (2014). Novel adaptive charged system search algorithm for optimal tuning of fuzzy controllers. *Expert Systems with Applications*, *41*(4), 1168–1175.

Ramírez-Ortegón, M. A., Märgner, V., Cuevas, E., & Rojas, R. (2013). An optimization for binarization methods by removing binary artifacts. *Pattern Recognition Letters*, *34*(11), 1299–1306.

Reed, R. (1993). Pruning algorithms-a survey. *IEEE Transactions on Neural Networks*, *4*(5), 740–747.

Rumelhart, D. E., Geoffrey, E. H., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*.

Santos, E., & Noggle, C. A. (2011). Synaptic Pruning. In *Encyclopedia of Child Behavior and Development* (pp. 1464–1465). Springer. Retrieved from http://link.springer.com/10.1007/978-0-387-79061-9_2856.

Scholkmann, F., Boss, J., & Wolf, M. (2012). An efficient algorithm for automatic peak detection in noisy periodic and quasi-periodic signals. *Algorithms*, *5*(4), 588–603.

Setiono, R., & Liu, H. (1995). Understanding neural networks via rule extraction. In *IJCAI* (Vol. 1, pp. 480–485). Citeseer. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.34.5143&rep=rep1&type=pdf.

Sharma, S. K., & Chandra, P. (2010). Constructive neural networks: a review. *International Journal of Engineering Science and Technology*, *12*(2), 7847–7855.

Sridhar, S. S., & Ponnavaikko, M. (2011). Improved adaptive learning algorithm for constructive neural networks. *International Journal of Computer and Electrical Engineering*, *3*(1), 30.

Stiles, J., Brown, T. T., Haist, F., & Jernigan, T. L. (2015). Brain and cognitive development. *Handbook of Child Psychology and Developmental Science*. Retrieved from http://onlinelibrary.wiley.com/doi/10.1002/9781118963418.childpsy202/full.

Stiles, J., & Jernigan, T. L. (2010). The basics of brain development. *Neuropsychology Review*, *20*(4), 327–348.

Sun, R. (2008). Introduction to computational cognitive modeling. *Cambridge Handbook of Computational Psychology*, 3–19.

UCI machine learning repository - Google Search. (n.d.). Retrieved July 21, 2016, from https://www.google.lk/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=uci%20machine%20learning%20repository.

Vollmer, L. (2010). Change Your Mind: Neuroplasticity & Buddhist Transformation. Retrieved from http://openscholarship.wustl.edu/cgi/viewcontent.cgi?article=1927&context=etd.

Wagarachchi, N. M., & Karunananda, A. S. (2013a). Novel technique for optimizing the hidden layer architecture in artificial neural networks. *American International Journal of Research in Science, Technology, Engineering and Mathematics*, 1–6.

Wagarachchi, N. M., & Karunananda, A. S. (2013b). Optimization of multi-layer artificial neural networks using delta values of hidden layers (pp. 80–86). IEEE. https://doi.org/10.1109/CCMB.2013.6609169.

Wagarachchi, N. M., & Karunananda, A. S. (2014). Towards a theoretical basis for modelling of hidden layer architecture in artificial neural networks. *Proc. of the Second Intl. Conf. on Advances In Computing, Electronics and Communication - ACEC*, 47–52.

Yu, X. H. (1992). Can backpropagation error surface not have local minima. *Neural Networks*, *3*, 1019–1021.